UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/728,370	12/03/2003	Glen Darling	50325-0841	6391

29989 7590 08/30/2007

HICKMAN PALERMO TRUONG & BECKER, LLP
2055 GATEWAY PLACE
SUITE 550
SAN JOSE, CA 95110

EXAMINER	
STEELMAN, MARY J	

ART UNIT	PAPER NUMBER
2191	

MAIL DATE	DELIVERY MODE
08/30/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/728,370	DARLING ET AL.	
	Examiner	Art Unit	
	MARY STEELMAN	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *21 June 2007*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *32-62* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *32-62* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *7/12/2007,8/6/2007*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This Office Action is in response to Remarks and Amendments received 06/21/2007. Per Applicant's request claims 32, 33, 37-39, 44-46, 51-53, 58, and 59 are amended. Claims 32-62 are pending. IDS received 7/12/2007 & 8/6/2007 has been considered.

Claim Rejections - 35 USC § 101

2. In view of the amendment to claim 51, the prior 35 U.S.C. 101 rejection is hereby withdrawn.

Response to Arguments

3. Applicant's arguments have been considered but are moot in view of the new grounds of rejection.

(A) Applicant has argued that (page 16, 1st paragraph) Bobick fails to teach or suggest "compiling source code files into one or more executable files."

Examiner maintains that Bobick discloses [0359], assets 240 are discovered in the source environment...creates an intermediate representation (compiled). The broadest reasonable interpretation of compiling is to change, and in this case source code is changed / grouped into an intermediate representation and packaged for deployment.

[0360-0365] In a preferred embodiment, an "export asset adapter method" (see FIG. 16) identifies and exports candidates for classification as assets 240 and together as packages

(compile assets into packages). In this preferred embodiment, the export asset adapter method 1600 (see FIG. 16 below) is used to obtain the actual current version of assets in the source tier that needs to be distributed to the target tier. After the assets are exported, the assets are moved to the deployment or distribution tier 960 and typically stored in an asset cache 961. When exporting assets, the export asset adapter method captures the logic, data, 210 and extended environment information 220 for an asset 240 and puts it into an asset data structure 240 where the asset type is also identified. [0361] the export asset adapter method (see FIG. 16B) takes the Asset Inventory 2100D and the intermediate representation 2100C and prepares a preliminary package specification 1650B. [0362] A package specification 1100 is created that in turn contains asset specifications...asset specification 1170 is stored in the deployment tier until a package 1100 (compiled package of assets) is scheduled to be delivered. [0363]- a packaging agent takes the preliminary package specification 1100A and creates a finalized package specification data structure [0365] If an asset requires processing the processing (compiling) may be done when the asset is stored in the asset cache 961 or at any time before the asset is distributed...The processing is primarily performed on the asset's extended environment, in an attempt to translate (compile) the extended environment to run in harmony with the base environment in the target tier...may also change the logic/data portion of the asset...

Additionally, Examiner points to compiling in the Forbes reference.

(B) Applicant argues that (page 17, 1st paragraph) Bobick fails to teach or suggest storing version information in the metadata of a module.

Bobick: [0223] Another exemplary embodiment and/or exemplary method of the present invention is directed to the discovery method, in which the asset definition data structure (metadata...versions) includes at least one of: an asset identification, an asset location, a URL, a name, an asset type, and a version. [0461] In another embodiment, a package structure may include an asset that is an asset adapter based on a CDS/ADS adapter asset type. The logic/data layer of this asset adapter asset may include an asset adapter class file supporting a particular asset type. The extended environment layer may contain versioning information. [0463] According to the example embodiment, each record or row 710 of the asset definition data structure 700 may contain a number of fields. An asset identifier field 720 may uniquely identify the asset for a particular application and may serve as the key or part of the key for the asset definition data structure 700...A version field 760 may identify the version or a time stamp for the asset and/or asset information. These aforementioned fields of the asset definition data structure 700 are exemplary. [0487] Any one of the assets in this package may have a version 1375 according to one embodiment. The version 1375 may be any known way of distinguishing the asset 1395. 'In a preferred embodiment, the version is a time stamp. Other examples of versions 1375 include locations, machine/node number, source, destination, checksum, differencing algorithm, file name, file system extended attributes, other versioning information, etc., or combination of these. While packages 1305 can include the actual assets 1395 of the package 1305, in a preferred embodiment, this is not done. Rather, some identifier of the asset 1395 may be included in a list.

(C) Applicant argues that (page 17, 2nd paragraph) Bobick fails to teach or suggest "collecting additional dependency information documented in one or more modules specifications."

Note Applicant's Specification recites, [0032], "A linker provides a list of dependent files for a given process or DLL, while the metadata documented in each dependent file lists the API names and versions the process or DLL depends on. [0033] The invention extracts explicit supplemental dependencies from the module specifications and encodes them in the module metadata just as it does with the linker generated data. [0095], Software packages contain package dependency information describing inter-node and intra-node package version dependencies. Specifically, each package defines which package versions it needs to be present locally and remotely. [0181], Explicit supplemental dependencies are documented in the module specification by the module author. The build environment collects these directly from the module specification and encodes them in the module metadata just as it does with the linker generated data. Explicitly defined and discovered dependencies are treated identically outside of the build environment.

Bobick disclosed: [0463] the asset definition data structure and a number of fields. An asset identifier field 720 may uniquely identify the asset for a particular application...A version field 760 may identify the version or a time stamp for the asset and/or asset information.

Bobick failed to explicitly disclose "additional dependency information documented in one or more modules specifications", in the sense of inter-node and intra-node package version

dependencies. (Examiner believes Applicant is referring to dependencies that dependent modules in the package have, i.e., 'nested dependencies.' Applicant's Specification [0095])

Forbes disclosed such 'nested dependencies' at [0013], [0045], [0060].

(D) It is noted on page 18, or Remarks, Applicant recites that claims 37, 44, 51, and 58 are independent claims which include some of the same features discussed above with respect to claim 32. However, all features for claim 32 are not included in claims 37, 44, and 51, but rather provided in further dependent claims 39, 46, and 53 respectively.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 32-62 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent Application Publication 2003/0172135A1 to Bobick et al., in view of US Patent Application Publication 2002/0144248 A1 to Forbes et al.

Per claims 32, 37, 44, 51, and 58:

A method of a development and build environment for packaged software delivery in a distributed network of nodes, the method comprising the computer-implemented steps of:

Bobick: [0068], packaging assets for processing and distribution over network...asset having a logic/data portion and an asset extended environment portion, and a package extended environment that includes package information associated with at least one asset.

-compiling source code files into executable file modules;

Bobick: [0339], An asset 200 may be defined as some meaningful partitioning of an application from the source environment...parts of the source environment that are selected to that the asset 200 can be distributed across a network. [0359], assets 240 are discovered in the source environment...creates an intermediate representation (compiled) [0360-0365] In a preferred embodiment, an "export asset adapter method" (see FIG.

16) identifies and exports candidates for classification as assets 240 and together as packages (compile assets into packages). In this preferred embodiment, the export asset adapter method 1600 (see FIG. 16 below) is used to obtain the actual current version of assets in the source tier that needs to be distributed to the target tier. After the assets are exported, the assets are moved

to the deployment or distribution tier 960 and typically stored in an asset cache 961. When exporting assets, the export asset adapter method captures the logic, data, 210 and extended environment information 220 for an asset 240 and puts it into an asset data structure 240 where the asset type is also identified. [0361] the export asset adapter method (see FIG. 16B) takes the Asset Inventory 2100D and the intermediate representation 2100C and prepares a preliminary package specification 1650B. [0362] A package specification 1100 is created that in turn contains asset specifications...asset specification 1170 is stored in the deployment tier until a package 1100 (compiled package of assets) is scheduled to be delivered. [0363]- a packaging agent takes the preliminary package specification 1100A and creates a finalized package specification data structure [0365] If an asset requires processing the processing (compiling) may be done when the asset is stored in the asset cache 961 or at any time before the asset is distributed...The processing is primarily performed on the asset's extended environment, in an attempt to translate (compile) the extended environment to run in harmony with the base environment in the target tier...may also change the logic/data portion of the asset... [0367], target processing asset adapter method 1900 (FIG. 19), targeted processing to change the Logic/Data section 210 of the asset data structure 240 in order to provide a unique asset...for one or more specific targets (compile assets for specific target).

Additionally, Examiner points to compiling in the Forbes reference. See Forbes [0085], where it is disclosed that an OSD (Open Software Description format) formatted manifest file can be embedded in an archive file, such as a JAVA Archive (JAR) file, or a composite compressed file,

such as a cabinet (.CAB) file, that contains the component's distribution unit to form a distribution unit file.

-wherein each of one or more modules contains an image for a process or a dynamically linked library (DLL);

Bobick: [0317], An asset may be a logical organization of information (e.g., software and data) that may serve as all or part of a package.

-creating a software package that comprises the one or more modules, wherein the software package is delivered to the nodes in the distributed network;

Bobick: [0317], A package structure may be composed of one or more assets. [0318], Various kinds of assets may be used in a package [0339], An asset 200 may be defined as some meaningful partitioning of an application from the source environment...parts of the source environment that are selected to that the asset 200 can be distributed across a network.

-wherein the software package is created based on at least one of a feature, characteristic, or purpose;

Bobick: [0325], Assets may be categorized by their purpose

-creating metadata for a first module, of one or more modules, that includes any module information such as the first module's: binary signature, name, directory path, and characteristics;

Bobick: [0360], extended environment information 220 for an asset 240, put into an asset data structure 240. [0372], The EE 220 (extended environment) has one or more common descriptors 210B, one or more asset dependency descriptors 222B, one or more target server dependencies 226B. [0373], Examples of common descriptors 210B include...a digital asset name...a unique fully qualified name...an address...a size...a volatility descriptor,...a security descriptor...

-inserting the metadata of the first module into the software package;

Bobick: [0372], The EE 220 (extended environment) has one or more common descriptors 210B, one or more asset dependency descriptors 222B, one or more target server dependencies 226B.

-gathering application program interface (API) dependency information for the first module, wherein the first module can provide and use at least one API, by

(a) receiving a list of dependent modules for a given process or DLL module of the first module;

Bobick: [0375], asset dependency descriptors

(b) storing, in the metadata of the first module, dependency information for the dependent modules in the list, wherein the dependency information includes API names and versions that the process or DLL module depends on;

Bobick: See FIG. 2B #220, Extended Environment & related text at [0377-0388], EE 220 has package relationship (dependency relationships) descriptors 285B [0396], reference descriptors 260B

Bobick: [0223], the discovery method, in which the asset definition data structure (metadata...versions) includes at least one of: an asset identification, an asset location, a URL, **a name, an asset type, and a version**. (emphasis added) [0461], a package structure may include an asset that is an asset adapter based on a CDS/ADS adapter asset type. The logic/data layer of this asset adapter asset may include an asset adapter class file supporting a particular asset type. **The extended environment layer may contain versioning information.** (emphasis added) [0463], each record or row 710 of the asset definition data structure 700 may contain a number of fields. An asset identifier field 720 may uniquely identify the asset for a particular application and may serve as the key or part of the key for the asset definition data structure 700...**A version field 760** (emphasis added) may identify the version or a time stamp for the asset and/or asset information. These aforementioned **fields of the asset definition data structure** (emphasis added) 700 are exemplary. [0487] Any one of the assets in this package may have a version 1375 according to one embodiment. The version 1375 may be any known way of distinguishing the asset 1395. In a preferred embodiment, the version is a time stamp. Other examples of versions 1375 include locations, machine/node number, source, destination, checksum, differencing algorithm, file name, file system extended attributes, other versioning information, etc., or combination of these. While packages 1305 can

include the actual assets 1395 of the package 1305, in a preferred embodiment, this is not done. Rather, some identifier of the asset 1395 may be included in a list.

(c) collecting additional dependency information documented in one or more additional modules specifications, wherein the additional dependency information includes API names and versions that the process or DLL module depends on;

Bobick: [0372], The EE 220 (extended environment) has...one or more asset dependency descriptors 222B (As an example, one or more (additional dependency information) [0463] the asset definition data structure and a number of fields. An asset identifier field 720 may uniquely identify the asset for a particular application...A version field 760 may identify the version or a time stamp for the asset and/or asset information.

More specifically Forbes:

[0013], the package manager acquires the manifest file and parses it to learn if the software package depends on any additional components. The package manager resolves any dependencies by acquiring a distribution unit containing the needed component...Because dependencies can nested within dependencies, the package manager recursively processes all the dependencies before finishing the installation of the software package that depends upon the additional components. [0019] The manifest file is not particular to a particular code type or operating system and allows for the specification of nested software dependencies. Because the manifest file contains the location of the distribution units for any dependencies... [0043] The names of other files in the distribution unit file 205, such as a text file containing licensing

information or a "readme" file containing special instructions for the software package, are listed in the manifest file 207 additional dependency information documented in one or more (additional modules specifications). The manifest file 207 also contains entries for software that is required to run (additional modules specifications) CoolestApp but which is not included in the distribution unit file 205. Such required software represent "dependencies" and frequently include such items as language libraries and common object class libraries (additional modules specifications). A dependency can also be another software package. The manifest file 207 provides the ability to describe the software dependencies in a recursive tree format, also known as a "directed graph." [0060], each dependency can itself include dependencies, the package manager will install all the nested dependencies prior to finishing the installation of the original software package.

(d) storing the additional dependency information in the metadata of the first module.

Bobick: [0329], EE 220 is a data structure (storage) containing descriptors and/ or other information required. [0463], each record or row 710 of the asset definition data structure 700 may contain a number of fields. An asset identifier field 720 may uniquely identify the asset for a particular application and may serve as the key or part of the key for the asset definition data structure 700...**A version field 760** (emphasis added) may identify the version or a time stamp for the asset and/or asset information. These aforementioned **fields of the asset definition data structure** (emphasis added) 700 are exemplary...While packages 1305 can include the actual assets 1395 of the package

1305, in a preferred embodiment, this is not done. Rather, some identifier of the asset 1395 may be included in a list.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention to modify Bobick's invention, using the teachings of Forbes, because Bobick recognized the need to (Bobick, [0061]) distribute data, programs, and portions of programs in a more efficient way over various tiers of a network to operate on any general platform or environment. Bobick disclosed (Bobick, [0068]) a system, method, and data structure for packaging assets for processing and distribution over a multi-tiered network. Likewise, Forbes disclosed (Forbes, [0007]) software distributed over the network, made to work across platforms or intelligently so that only the correct version of platform specific software is pushed down to the user. Forbes recognized the need for (Forbes, [0011]) a software distribution and tracking mechanism that handles cross-platform software, specifies the component dependencies, and is applicable to both the older distribution media as well as to the network distribution paradigm.

Per claims 33, 38, 45, 52, and 59:

-a linker creates the list of dependent modules for the given process or DLL module and places the list in the metadata of the first module.

Bobick: [0377-0379] EE 220 has package relationship descriptors 285B that represents a part-whole relationship between the digital asset 240 and one or more packages containing the digital asset. [0379], the specific package does not possess the property of correctness unless the specific digital asset is included...this relationship can further imply the existence of the other

digital assets that are members of the specific package (list of dependent modules for given process or DLL) [0383], The EE 220 includes one or more base environment descriptors 225B (e.g., target server dependencies descriptors 226B) (dependent modules) that identify a base execution environment on one or more target computers. The base execution environment 250 is required to execute the digital asset... [0433] FIG. 6 package data structure showing the assets associated with a package...identifies the assets that are grouped together in a package

Per claims 34, 40, 47, 54, and 60:

-creating metadata for each API;

-inserting the API metadata into the software package, wherein metadata for an API includes, but is not limited to: the API's name and version.

Bobick: Creating EE (metadata) for each asset (see above). Creating package structures ([0433+], #600), example API's disclosed at [0438]: a Web server for an SC asset, a Java servlet engine for a JSP, a Java Runtime Environment for a Java class asset, and application server for an EJB asset...DBMS for data assets, a Minimum Application Server (MAS) may be used in the base environment. The MAS may provide the minimal services that an application when distributed to a client (application programming interfaces). Also see APIs at [0439], adapters: a discovery adapter, a versioning adapter, an export adapter, a process adapter, a target adapter, a client deployment adapter, a synchronization adapter, a bridging adapter, an adjustments adapter, a streaming adapter, a quality of service adapter...

[0466], asset identifier field 820 (name) may uniquely identify the asset...version field 830

[0488], FIG. 14, package definition data structure 1400, package identifier filed package ID, name and / or any other information that may uniquely identify a packager such as, for example, a package number (version)

Per claims 35, 41, 48, 55, and 61:

-calculating a binary signature for each module of the one or more modules and inserting the binary signature into the respective module's metadata;

wherein each unique version of a module will have a unique binary signature.

Bobick: [0401], security descriptors (binary signatures)

Per claims 36, 43, 50, 57, and 62:

-creating metadata for the software package that includes any package information such as the software package's: name, build date, and characteristics;

-inserting the metadata of the software package into the software package.

Bobick: engagement table 100H may contain a plurality of system part 120H to target node 130H pairs...Each engagement pair may contain a unique part identifier...name...address and / or any other identifier capable of uniquely identifying (Each asset for the software package includes information placed in the package.) [0327], A package may be categorized by type and / or a set of types...may include one or more assets...may have a package type...An asset type is an identifier of an asset...may also determine what information that the asset, and hence the

package, need to execute on any given remote target environment. A package specification includes a description of the package structure (metadata) including the package types.

Per claims 39, 46, and 53 :

-collecting additional dependencies documented in one or more additional module specifications and placing them into the metadata of the first module;

wherein the additional dependencies documented in each module lists API names and versions that the process or DLL module depends on.

See limitations addressed in claim 1 above. [0354], a package structure may contain one or more assets...

Per claims 42, 49, and 56:

-packages are created based on at least one of a feature, characteristic, or purpose.

Bobick: [0340], An asset may be categorized by the content and / or purpose...

Conclusion

6. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after

the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Mary Steelman, whose telephone number is (571) 272-3704. The examiner can normally be reached Monday through Thursday, from 7:00 AM to 5:30 PM If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Zhen can be reached at (571) 272-3708. The fax phone number for the organization where this application or proceeding is assigned: 571-273-8300.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Mary Steelman

08/23/2007

Conclusion

8. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Mary Steelman, whose telephone number is (571) 272-3704. The examiner can normally be reached Monday through Thursday, from 7:00 AM to 5:30 PM If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Zhen can be reached at (571) 272-3708. The fax phone number for the organization where this application or proceeding is assigned: 571-273-8300.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Mary Steelman

08/20/2007

MARY STEELMAN
PRIMARY EXAMINER